REPORT R-399 DECEMBER, 1968

CSL COORDINATED SCIENCE LABORATORY

SYNTACTIC ANALYSIS FOR THE R2 SYSTEM

KENNETH BISS

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

This work was supported by the U.S. Office of Education under Contract OE C-1-7-071213-4557; auxiliary support was provided by the Joint Services Electronics Program (U. S. Army, U.S. Navy, and U. S. Air Force) under Contract DAAB 07-67-C-0199.

ACKNOWLEDGMENT

I wish to express my gratitude to Professor R. T. Chien for his invaluable advice and guidance throughout the period of research for this report. Thanks are also extended to Jeff Schultz for supervising the construction of the dictionary for me and to Mrs. Julie Anson for the preparation of this document.

TABLE OF CONTENTS

1. INTRODUCTION

The R2 system is being designed to automatically answer questions on the Illinois Driver's Manual--Rules of the Road. That is, the R2 system will accept any question based on Rules of the Road and, through some search procedure, pick the answer to the question. At some point in the search procedure it will be necessary to syntactically analyze the input question or some data text, or both. It is for this reason that we initiated the investigation which has led to this report.

In our search for a syntactic analysis program for the R2 system we came across many programs which were written in the last few years. Since we were interested in efficiency as well as effectiveness we studied the more basic immediate constituent analysis and dependency analysis leaving the transformational grammar programs for some later date.

In Sections Two through Five we will discuss some of the grammars and associated programs that were considered for our syntactic analysis program.

In Section Two we discuss phrase structure grammars and standard form phrase structure grammars showing how these grammars are used to generate the sentences of a language.

In Section Three we discuss immediate constituent analysis which is the analysis procedure usually associated with phrase structure

grammars and the predictive analysis program of Kuno[10,11,12] which is based on a standard form grammar.

In Section Four we discuss dependency grammars and its relation to phrase structure grammars.

In Section Five we discuss some dependency grammar analysis programs, and finally, in Section Six, we discuss the result of our search for an analysis program, namely our own dependency grammar parsing program. This program is now operating on the CDC 1604.

2. PHRASE-STRUCTURE GRAMMAR

It is not known exactly how the speaker of a language picks a sentence to communicate an idea he wishes to express. In some way he picks a group of words from the vocabulary of the language he wishes to communicate in, and a structure which gives the relationships among the words he has picked. He then puts the words and the structure together, putting the words in an order which reflects the structure and modifying words where it is necessary to further reflect the structure. The listener then takes note of the words the speaker used, the order he used them in, and the modifications he made to the words. From this information the listener knows what idea the speaker was trying to communicate.

The rules which a listener uses to discover the underlying structure of the speaker's sentence are called the grammar of the language. These rules enable the listener to associate a structure with each sentence he hears, a structure which should be relatively independent of the lexical entries in the sentence.

The rules which the listener uses to parse sentences are the inverse of the rules which the speaker uses to generate sentences. Thus if we understand the process of sentence generation we should understand how to parse sentences. So let us now look at some grammars which enable us to generate sentences.

A context free phrase structure grammar consists of a non-terminal alphabet (e.g. NP-noun phrase, VP-verb phrase, etc.), a terminal alphabet (e.g. noun, verb, etc.) and a set of rewrite rules. These rewrite rules are of the form S → NP VP PD which means whenever we encounter the symbol S we can replace it with the concatenated triple NP VP PD. For example, we might have the grammar with the following rules:

1) S → NP VP PD
2) NP' → A_j N
3) PP' → P NP'
4) NP' → T N
5) VP → V PP'
6) NP → T NP'
7) A_j → adj
8) V → verb
9) N → noun
10) T → def
11) T → indef
12) P → prep
13) PD → pd

To generate sentences from the above grammar we successively apply the rules being sure to start the process with a rule whose left side is the initial symbol S. We can thus generate the sentence, "The old man sat on the bench." as is done on the next page.

S

NP VP PD	rule 1
T NP' VP PD	rule 6
def NP' VP PD	rule 10
def A_j N VP PD	rule 2
def adj noun VP PD	rule 7,9
def adj noun V PP'PD	rule 5
def adj noun verb PP'PD	rule 8
def adj noun verb P NP' PD	rule 3
def ajd noun verb prep NP' PD	rule 12
def adj noun prep TN PD	rule 4
def adj noun verb prep def noun period	rule 10,9,13

(the)(old)(man)(sat)(on)(the)(bench).

The above derivation can also be represented as a tree (fig. 1).

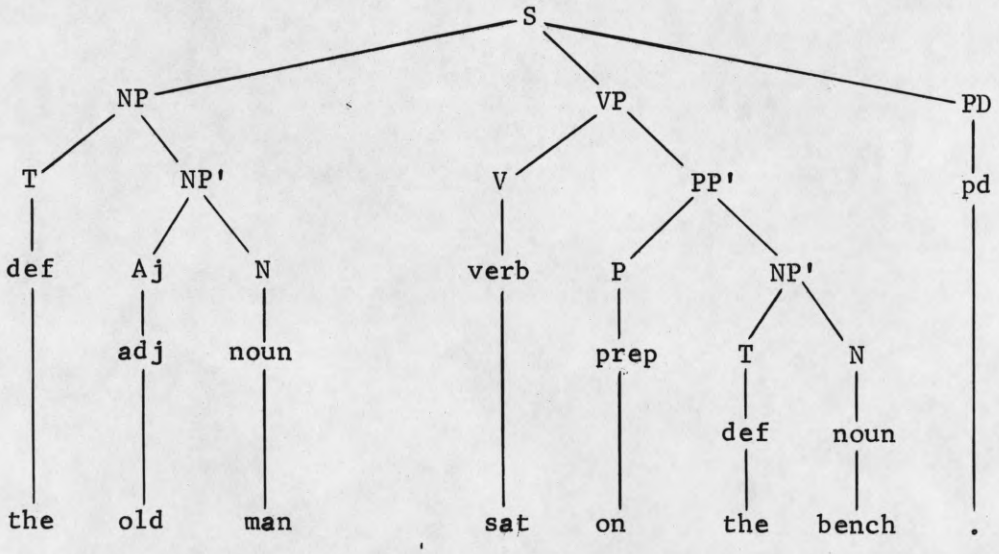

Figure 1

If we slightly modify the above rules we can form a standard form grammar whose rules are the following:

1) S → def NP VP PD

1a) S → indef NP VP PD

2) NP → adj N

3) N → noun

4) VP → verb PP'

5) PP' → prep NP

6) NP → def N

6a) NP → indef N

7) PD → pd

Let us again generate "The old man sat on the bench." this time using the standard form grammar. This derivation is shown below and is represented by the tree of Figure 2.

S

def NP VP PD	rule 1
def adj N VP PD	rule 2
def adj noun VP PD	rule 3
def adj noun verb PP' PD	rule 4
def adj noun verb prep NP PD	rule 5
def adj noun verb prep def N PD	rule 6
def adj noun verb prep def noun PD	rule 3
def adj noun verb prep def noun pd	rule 7

(the)(old)(man)(sat)(on)(the)(bench).

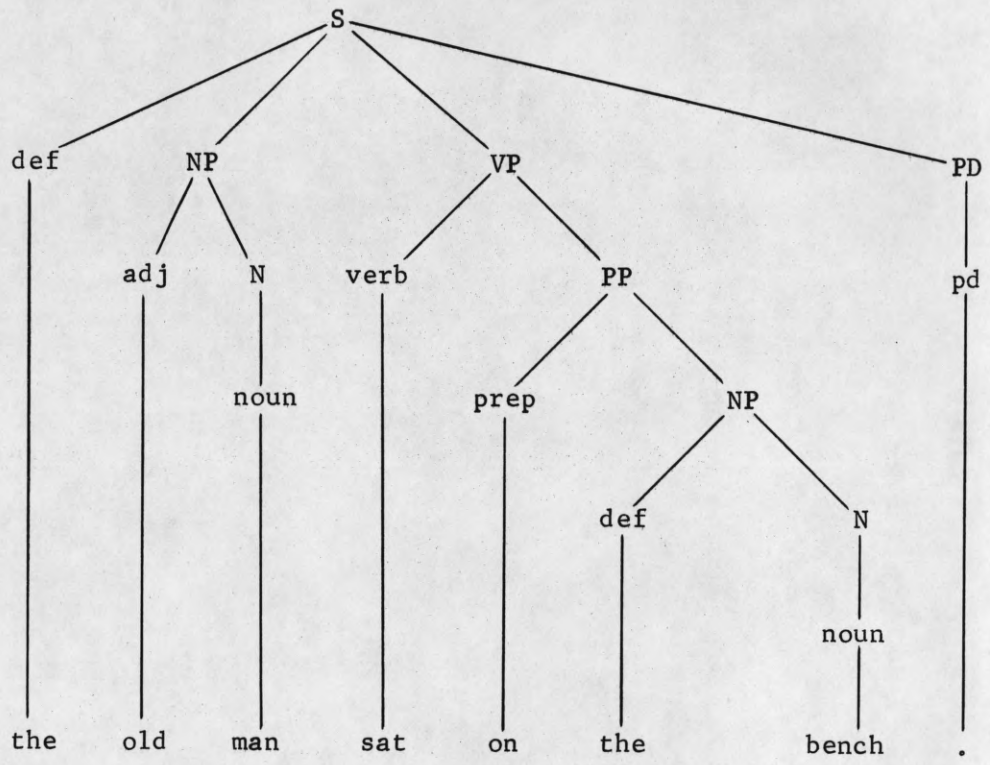
S
def
NP
VP
PD
adj
N
verb
PP
pd
noun
prep
NP
def
N
noun
the
old
man
sat
on
the
bench
.

Figure 2

3. SOME PHRASE-STRUCTURE PARSING ALGORITHMS

Immediate constituent analysis is the inverse of context-free phrase structure sentence generation.

Instead of having rules which allow us to expand terms we have rules which allow us to combine terms. That is, the rewrite rules for an immediate constituent analysis are of the form Aj N → NP', which states that an adjective and a noun which appear next to each other in a sentence may be combined to form a term called an NP'. To analyze a sentence we first find the syntactic class of each word in the input string. Using the string of syntactic classes we search for adjacent words which the rules indicate can be combined. We put parentheses around the combined words and treat this parenthetic expression (p.e.) as a syntactic word class and search for new combinations.

Let us apply the idea of immediate constituent analysis to the sentence "The old man sat on the bench." assuming the rules of our analyses are the following:

1) adj noun → NP
2) def noun → NP'
3) NP VP PD → S
4) prep NP' → PP'
5) verb PP' → VP
6) def NP' → NP

The analysis would then be:

The old man sat on (the bench)	rule 2
The (old man) sat on (the bench)	rule 1
The (old man) sat (on (the bench))	rule 4
(The (old man)) sat (on (the bench))	rule 6
(The (old man)) (sat (on (the bench)))	rule 5
[(The (old man)) (sat (on (the bench)))]	rule 3

Predictive analysis as used by Kuno[10,11,12] is the inverse of standard form sentence generation. The rules used in the analysis are of the form (S, def)/NP VP PD. This rule is a prediction that if a sentence starts with "def" it will be followed by NP VP PD. Some of the rules will be of the form (N,noun)/Ω, Ω being the empty string.

The analysis starts by finding the syntactic classes of the words in the input string and putting the initial symbol S into a push down store (PDS). The machine then looks at the first word of the input string and tries to find a rule whose left side is (S, first word in input string). If no such rule is found the string is ill-formed. If a rule is found, the symbols of the right hand side of the rule replace S in the PDS and the analysis goes to the second word.

At the n^{th} step in the analysis, the symbol at the top of the PDS is compared with the n^{th} input symbol. If a rule whose left hand side is of the form (symbol at top of PDS, n^{th} input symbol) is not found, the string is ill-formed. If such a rule is found the right hand side of this rule replaces the symbol at the top of PDS and the analysis goes to the $(n+1)^{th}$ input symbol. If processing the last input symbol yields an empty PDS, we have a parsing.

As an example of predictive analysis let us parse the by-now familiar sentence "The old man sat on the bench," assuming the rules of our analysis system are as follows:

1) (S, def)/NP VP PD

1a) (S, indef)/NP VP PD

2) (NP, adj)/N

3) (N, noun)/Ω

4) (VP, verb)/PP'

5) (PP', prep)/NP

6) (NP, def)/N

6a) (NP, indef)/N

The analysis would be:

The old man sat on the bench

(def)(adj)(noun)(verb)(prep)(def)(noun)

contents of PDS

1) S

2) NP
   VP replacing S rule 1
   PD

3) N
   VP rule 2
   PD

4) VP rule 3
   PD

5) PP' rule 4

PD

6) NP rule 5

PD

7) N rule 6

PD

8) PD rule 3

9) Ω rule 7

To form the phrase structure tree which underlies this sentence we simply recall the rules which led us to step nine.

4. DEPENDENCY GRAMMARS

We say that word A depends on word B, or that B governs A, if A modifies or complements the meaning of B in a sentence. A grammar which is based on the idea of dependents and governors is called a dependency grammar.

Let us now summarize the formal theory of dependency grammars that Hays[8] has developed.

A dependency grammar consists of a non-terminal alphabet (e.g. Npl, -Ving, etc.), a terminal alphabet whose members are words or morphemes, and a set of dependency rules. The rules are of the form $X_i(X_{i_1}, X_{i_2}, \ldots, X_{i_k}, *, X_{i_j}, \ldots, X_{i_n})$ where the X_{i_m} belong to the non-terminal alphabet. This rule states that X_i governs $X_{i_1}, \ldots, X_{i_n}$, that X_{i_1} will appear ahead of X_{i_2}, X_{i_2} will appear ahead of X_{i_3}, and that X_i will appear between X_{i_k} and X_{i_j}. A rule of the form $*(X_k)$ indicates that X_k need not have a governor, while a rule of the form $X_j(*)$ indicates that X_j need not have any dependents.

To generate a sentence we first pick an element which appears in a rule of form $*(X_k)$. We then find rules of the form $X_k(X_{k_1}, \ldots, *, \ldots, X_{k_n})$ and attach the dependents of X_K to X_k. We now search for rules which indicate dependents of the X_{k_i} and make the dependency connections. We continue this process until we assign, to all units which have no dependents, a rule of the form $X_m(*)$. This completes the first step of the derivation. The second step of the derivation procedure replaces the non-terminal symbols with terminal symbols.

The process can be thought of as building a tree. The nodes of the tree are the symbols (first nonterminal, then replaced by terminal symbols). The connections between the nodes indicate dependency (i.e., if two words are directly connected in the tree, the lower node depends on the higher node). The independent symbol is the head of the tree, and the elements which appeared in rules of the form $X_m(*)$ are at the bottom of the tree.

As a simple example let us derive the sentence "The old man sat on the bench," assuming we have the following rules:

1) I(V)

2) N (T, Aj, *)

3) N (T, *)

4) V (N, *, P)

5) T (*)

6) Aj (*)

7) P (*, N)

And assuming we have a function which carries out the following mapping:

V → sat, hit, run, etc.

N → man, woman, bench, house, etc.

P → on, in, etc.

T → the, a, etc.

Aj → old, new, large, etc.

Applying the rules we get the following derivation:

*(V)	rule 1
*(V(N, *, P))	rule 4

*(V (N (T, Aj, *), (, P)) rule 2

*(V (N (T (*), Aj, *), *, P)) rule 5

*(V (N (T (*), Aj (*), *), *, P(*, N))) rule 7

*(V (N (T (*), Aj (*), *), P (*, N (T, *)))) rule 3

*(V (N (T (*), Aj (*), *), *, P(*, N (T (*), *)))) rule 5

This is equivalent to the tree of Figure 3.

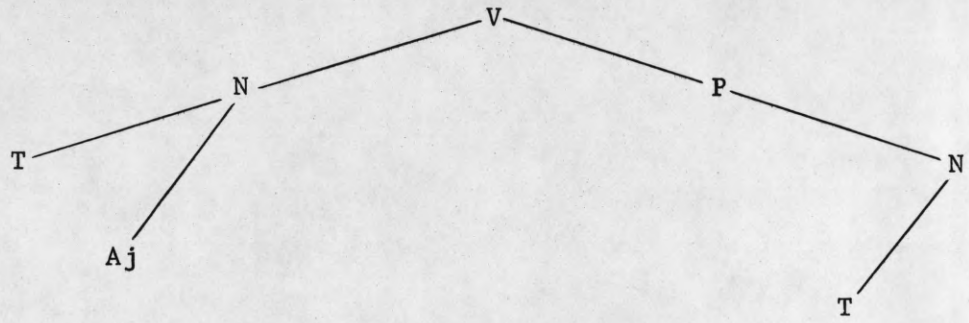

Figure 3

Now applying the function to the tree of Figure 3 we get the tree of Figure 4.

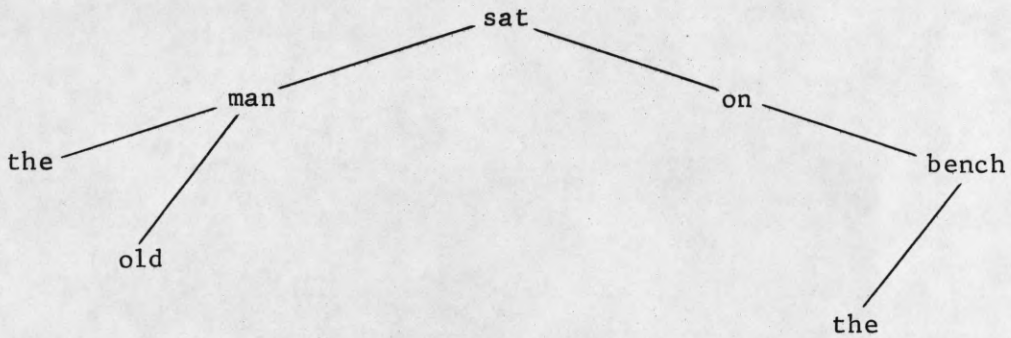

Figure 4

Let us look more closely at the tree in Figure 4. If we project a line down from each node to a base line, we see that no projection line crosses a dependency link (Figure 5--the dashed lines are the projection lines).

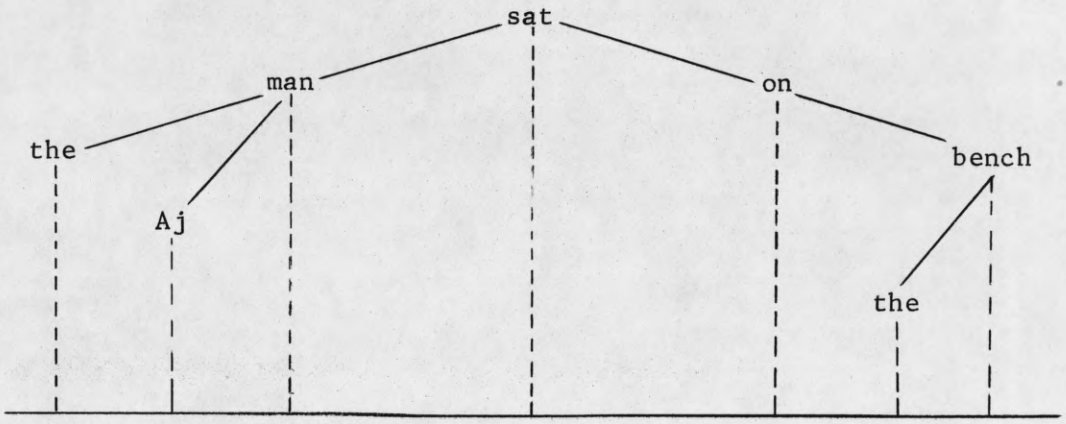

Figure 5

This properly is called projectivity and any sentence which satisfies this property is a projective sentence. Not all English sentences satisfy projectivity as Hays[6] points out. An example due to Hays of a non-projective sentence appears in Figure 6. The point where the projection line and the dependency link cross is circled.

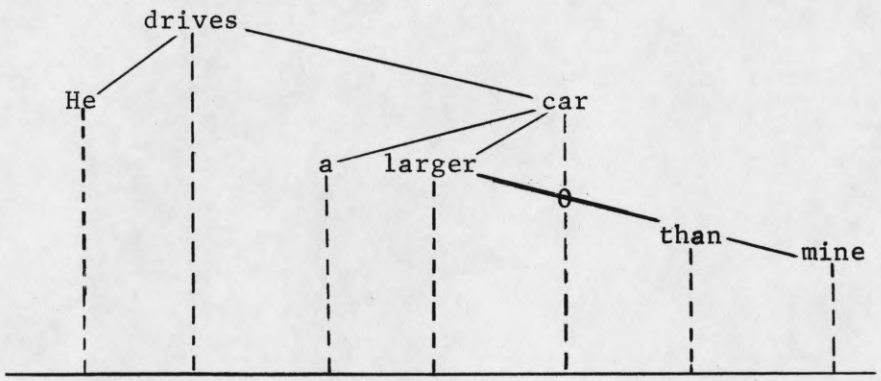

Figure 6

It should be noted that an immediate constituent generation of the sentence in Figure 6 would show no relation between larger and than. Thus non-projective sentences are made up of non-immediate constituents. Thus the assumption that a sentence is projective is the same as assuming that the sentence is made up of immediate constituents.

Projectivity is equivalent to the statement: any word that occurs between two connected occurrences in a sentence must depend on either one of those words or the other.

Proof: Assume two words (X, Y) are connected in a projective sentence. Assume Z lies between X and Y in the sentence and assume Z does not depend on X or Y. Then Z must depend on some occurrence to the left of X or to the right of Y or Z is independent. If Z depends on something to the left of X then its dependency connection will cross the projection line of X (Figure 7).

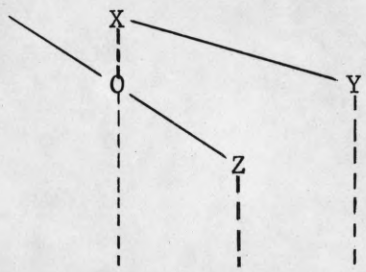

Figure 7

If Z depends on something to the right of Y then Z's dependency connection crosses Y's projection line (Figure 8). If Z is independent then its projection line crosses the dependency connection between X and Y (Figure 9).

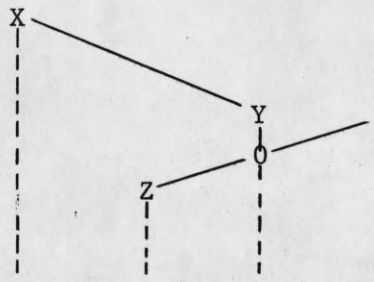

Figure 8

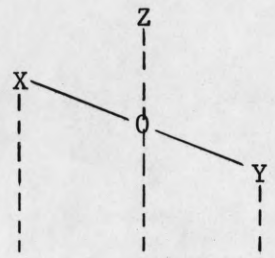

Figure 9

Before we give some examples of dependency parsing programs, let us see the relationship between projective dependency trees and immediate constituent analysis.

Hays[7] has shown how we can derive a dependency parsing for a sentence given an immediate constituent parsing and vice versa. Let us now summarize how this is done.

Given a projective dependency tree let us derive an immediate

constituent analysis (i.e., a parenthetic expression).

First we embed the tree in a plane and drop projection lines from each node of the tree to the base line. Now insert parentheses around all points that are projected from complete subtrees and make all possible permutations of these p.e.'s.

If we now have p.e.'s with at least three other p.e.'s contained in them we can form optional p.e.'s. Consider a p.e.: we will call it A, which has $N \geq 3$ p.e.'s contained in it. A is the projection of a complete subtree. Now we take any 2, 3, ..., N-1 of the p.e.'s within A, including the head of the complete subtree which projects into A, and take any permutation of this new set of p.e.'s, making sure we keep the constituents contiguous. We can continue this process as long as we still have p.e.'s with at least three p.e.'s contained in them.

Let us look at an example:

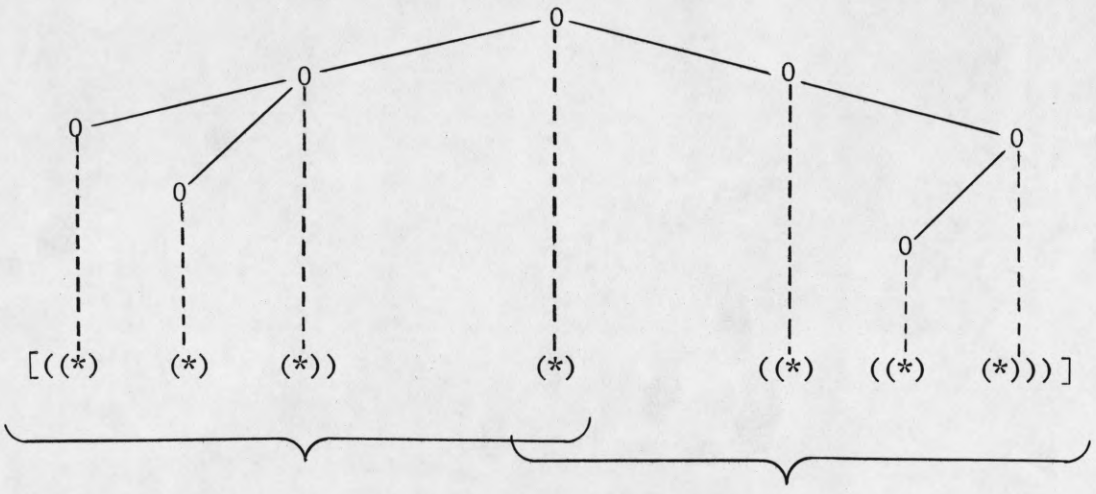

Figure 10

Figure 10 shows the transformation from the tree to an immediate constituent analysis with the optional p.e.'s in brackets.

Now let us see how to go from a p.e. to a dependency tree.

First we strip away the outermost parenthesis leaving two or more p.e.'s. Choose one of these and make it the head of a tree with the others dependent on it. If every node corresponds to an elementary p.e. then we are done. If not, look at each node which corresponds to a non-elementary p.e. Drop the outermost parenthesis of the p.e. thus leaving two or more p.e.'s. Choose one of these to correspond to the given node. Continue this process until all nodes correspond to elementary p.e.'s.

As an example, let us reverse the procedure used in Figure 10. Figure 11 shows two dependency trees which were derived from the given parenthetic expression.

[((*)(*)(*)) ((*) ((*)((*)(*))))]

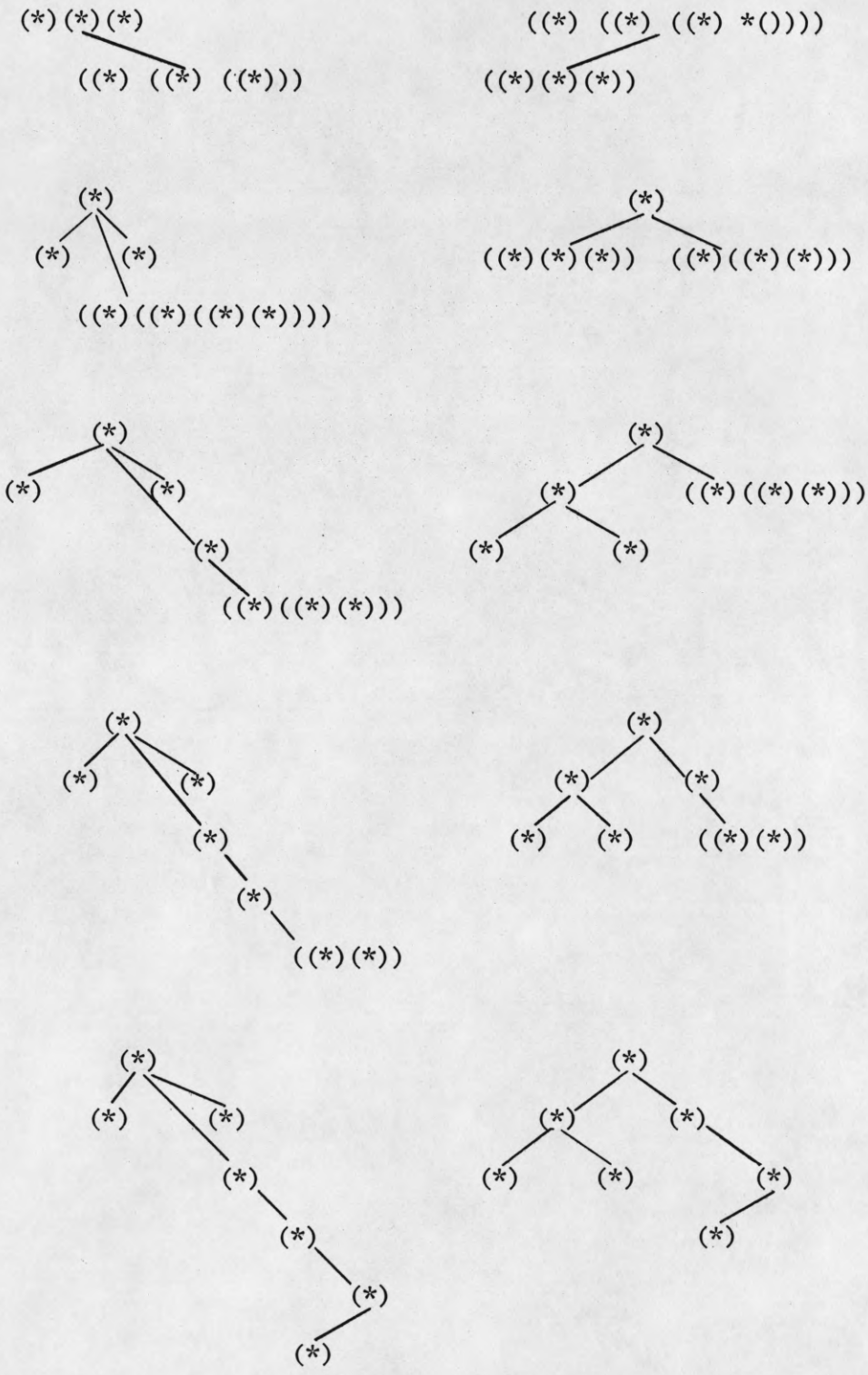

Figure 11

5. SOME DEPENDENCY PARSING PROGRAMS

It is now obvious that dependency analysis and immediate constituent analysis are related. In fact, Gaifman[5] has shown that dependency grammars and immediate constituent grammars are weakly equivalent. That is, they generate the same set of sentences from the same initial vocabulary or, analytically, they classify the same set of sentences. However, dependency grammars and immediate constituent grammars are not strongly equivalent, which means that there is no isomorphism between the structural diagrams each associates with a given sentence. Thus, there will be some sentences which are ambiguous to immediate constituent grammars (i.e., there will be several structural descriptions assigned to the sentence) but unambiguous to dependency grammars, and vice versa. This means, as Hays[7] points out, that immediate constituent grammars obtain something of syntax which dependency grammars miss, and vice versa.

Let us now look at some parsing programs which have used dependency grammars. A dependency parsing consists of finding the dependency tree for the sentence being parsed. The main verb of the sentence is usually taken as the head node of the tree with the other words of the sentence appearing at the other nodes.

Hays[9] has written a dependency grammar parsing program as part of machine translation system. The first step in the program is to

assign every word in the input sentence its grammar code symbol; this is done by dictionary lookup. Using the string of grammar code symbols the program checks all possible pairs of words which could be connected in a projective sentence for a dependency relation. We say that two words could be connected in a projective sentence if they satisfy precedence where precedence is defined in the following way.

x precedes y iff

1) x appears to the left of y in the input sentence.
2) Any word that is between x and y depends on either x or y (this is to insure projectivity)
3) Neither x depends (directly or indirectly) on y or y depends on x (this is to insure no circular dependency connections).
4) Either x or y or both are independent (this is to insure that all elements in the sentence have only one governor).

When a pair of words are tested for and become a precedence pair, we can test the pair for a dependency connection. We will allow x to depend on y if

1) x is independent
2) x precedes y or y preceds x
3) there is an entry in the table of dependency types which states that the grammar code symbol of x can depend on the grammar code symbol of y.

If the above three conditions are satisfied we make the connection between x and y then modify the grammar code symbols of both x and y to

show the properties of the pair x,y. This modification turns the x,y pair into a constituent of an immediate constituent analysis. In this way Hays obtained an immediate constituent analysis along with the dependency analysis. The final output of Hay's program is all possible dependency parsings of the input sentence compatable with the grammar of the system, along with the associated immediate constituent analysis. Hays decided to give both immediate constituent parsings and dependency parsings because, as we pointed out a few pages ago, each captures something of grammar which the other misses.

Simmons[13] has also written a dependency parsings program called the "Pattern-Learning Parser"(P-LP) which "learns" the grammar rules used in the grammar from a set of hand-parsed sentences.

To be more specific, a set of sentences are hand parsed and each word in the sentences is assigned a number to show at what level it appeared in the dependency tree. For example:

```
.  "The old man sat on the bench."
0   -3  -3  -2  +1  +2 -4    +3 0
```

would be assigned the numbers which appear below the sentence (note that the numbers on the object side of the verb are positive). The P-LP then forms a word class for each word encountered in the hand parsed text, and a set of sentence pattern rules which indicate the structure of the dependency trees of the hand parsed text.

The word class for a particular word is of the form a/b/c where "a" is the depth in a dependency tree at which the particular word occurred, "b" is the depth at which the word to the left of "a" occurred,

show the properties of the pair x,y. This modification turns the x,y pair into a constituent of an immediate constituent analysis. In this way Hays obtained an immediate constituent analysis along with the dependency analysis. The final output of Hay's program is all possible dependency parsings of the input sentence compatable with the grammar of the system, along with the associated immediate constituent analysis. Hays decided to give both immediate constituent parsings and dependency parsings because, as we pointed out a few pages ago, each captures something of grammar which the other misses.

Simmons[13] has also written a dependency parsings program called the "Pattern-Learning Parser"(P-LP) which "learns" the grammar rules used in the grammar from a set of hand-parsed sentences.

To be more specific, a set of sentences are hand parsed and each word in the sentences is assigned a number to show at what level it appeared in the dependency tree. For example:

```
.  "The old man sat on the bench."
0   -3  -3  -2  +1  +2 -4    +3 0
```

would be assigned the numbers which appear below the sentence (note that the numbers on the object side of the verb are positive). The P-LP then forms a word class for each word encountered in the hand parsed text, and a set of sentence pattern rules which indicate the structure of the dependency trees of the hand parsed text.

The word class for a particular word is of the form a/b/c where "a" is the depth in a dependency tree at which the particular word occurred, "b" is the depth at which the word to the left of "a" occurred,

and "c" is the depth at which the word to the right of "a" occurred. If the word has occurred in several places of the hand parsed text then all of the word classes which have been assigned to the word are logically added. This yields an expanded word class of the "a,b,c/d,e,f/g,h,j where a,b,c are the depths at which the word has occurred in the various trees, etc.

The sentence pattern rules formed from a hand parsed sentence indicate the structure of the dependency tree of the sentence above a certain level. For example, the word "The" of the sentence "The old man sat on the bench" generates the rule "-3/-3/-2/+1/+3/+3/0/0," "man" generates '2/+1/+2/00" and "bench" generates "+3/0/-2/+1/+2/0" (here the zeros are used as directional markers). When a sentence is given to the P-LP to parse, the program first finds, for each word in the input string, a word class. Then certain parts of adjacent words word classes are logically added. This logical addition allows the words to the left and right of the word in question to predict at what level the word will occur.

The new set of word classes for each word are then checked against a table of possible triads which indicate what 3-tuples of the form a/b/c were encountered in the sample text. If a particular triad was never encountered in the sample text then it is not allowed to occur in the word classes of the input sentence. To help reduce ambiguity at this stage, certain parts of the word classes are logically subtracted with their left and right neighbors (this is the reverse of the logical addition which was previously done).

The altered word class of each word of the sentence is now intersected with the original word class assigned to the word. If the intersection leaves the word class unchanged or reduces it to zero, no change is made in the words altered word class. If the class is changed by the intersection, the result of the intersection replaces the altered word class (this gives greater weight to the experienced depth code over neighbors predictions). The words of the input sentence now should be almost unambiguously assigned a depth in the dependency tree. The governor is easily found since a word at level-3 requires a governor at level ± 2, etc. If after looking for all governors for all words of the sentence ambiguity still exists, we apply the sentence pattern rules.

The P-LP outputs multiple parsings of a given input sentence of which the first parsing is chosen.

6. A DEPENDENCY PARSING PROGRAM FOR THE R2 SYSTEM

After careful consideration we decided to take a different approach than that suggested by Hays' and Simmons' programs. We decided not to take Hays' approach for two reasons. First, we felt that for the R2 system we did not need to impose the restriction that all sentences must be parsed as projective sentences. Second, we felt that we could get useful results by obtaining the most probable parsing for a given sentence. By allowing only one parsing we can save time in the program and we can eliminate the problems associated with multiple parsings.

We decided not to take Simmons' approach also for two reasons. First, we felt that the rules which the P-LP generates might be too general for the R2 system. Second, as we stated above, we wanted a program which would select the most probable parsing for any sentence.

We have written a dependency parsing program which will eventually be integrated into the R2 system. The program is written in CSL6 (the local form of the L^6 Language) for programming ease.

We chose to use a dependency grammar parser for two reasons. First, it seemed to us that dependency analysis would lend itself to fast running, easily written programs. Second, it appeared to us that a dependency grammar would give us the most revealing structural description of the sentences in Rules of the Road.

Let us now explain our program. The program uses two tables. The first table is a dictionary containing words and their syntactic word classes. Any word is allowed to be a member of up to three syntactic classes. The second table is a table of grammar rules. Following Hays, this table lists the syntactic class of a dependent, the syntactic class which may govern the symtactic class of the dependent, and a number which indicates the order which the governor and its dependent must have (i.e., the entry is two if the dependent must be to the left of the governor, one if the governor is to the left of the dependent, and zero if there is no order restriction in the rule).

The input to the program is the words of the sentence to be parsed. The first step is to assign the syntactic classes to all the words of the input sentence; this is done by dictionary lookup. The program is now ready to start the actual parsing.

Initially, the first word of the input string is considered a dependent and the program searches for its governor. We first test the word to the immediate right of this particular word, then the second word to the right, etc., until we find a governor for the first word. To test for a dependent-governor pair we go to the table of rules. If the class of the first word of the sentence is the first word of a rule and the class of the candidate governor is the second word of a rule, and if the word order required by the rule is satisfied, then the first word is connected to this candidate governor and we go to the second word of the sentence. If we are at the k^{th} word of the sentence we consider the k^{th} word as a dependent and then check the words at distance

one from the dependent for governorship, checking the $(k+1)^{th}$ word first, then the $(k-1)^{th}$ word. If neither of these is the governor we go to a distance two from the k^{th} word, checking to the right first, then to the left. If no governor is found we continue to search further away from the k^{th} word until either the governor of the k^{th} word is found or until we have checked all other words in the sentence. If no governor is found the word is taken to be the head of the dependency tree underlying the sentence. If the k^{th} word has more than one syntactic class associated with it, then we search for a governor for each syntactic class. That is, if word j has syntactic classes "A" and "B" associated with it, we first search for a governor for the j^{th} word as above, assuming the j^{th} word is a member of syntactic class "A." Once this governor is found we start again at the j^{th} word, now assuming that the word is a member of syntactic class "B." If in searching for the governor of the m^{th} word we came across a governor candidate which has two or three syntactic classes associated with it, we first test for governorship assuming the governor candidate is a member of the first class assigned to it. If a dependency connection is made we go to the $(m+1)^{th}$ word and start over looking now for a governor of the $(m+1)^{th}$ word. If a dependency connection is not made we test the second syntactic class of the governor candidate for agreement with the m^{th} word. That is, if in searching for the governor of the m^{th} word we encounter the n^{th} word which has classes "C" and "D" associated with it, we check the m^{th} and the n^{th} word for agreement assuming the n^{th} word is a member of class "C." If a connection is made in this way

between the m^{th} and n^{th} words, we go to the $(m+1)^{th}$ word and start searching for its governor. If no connection is made we check for agreement between the m^{th} and n^{th} words assuming the n^{th} word is a member of syntactic class "D." If a connection is then made we go to the $(m+1)^{th}$ word and search for its governor. If no connection is made we leave the n^{th} word and search for the governor of the m^{th} word elsewhere.

Be searching in this fashion we emphasize that the governor for any word is most likely to occur very near the particular word in question.

To test our program we first compiled a dictionary containing all the words from Rules of the Road. We then selected a small set of simple, compound and complex sentences from the driver's manual, selecting several sentences from each chapter of the manual.

The results of this test indicate that the parsing we get has a fairly high probability of being the correct parsing. If this parsing is the wrong parsing, then we could attach dependents to the second candidate governor instead of the first. This would give another parsing keeping with the philosophy that dependents are close to their governors; although in general, this parsing would have a lower probability of being correct than the first parsing. This is, in fact, what we intend to do in the future. More precisely we will form a sequential procedure which employs a semantic-syntactic loop in which the parsings of a sentence will be ranked according to their probability of being correct. When a sentence is presented to this semantic-syntactic loop system it will be assigned its most probable parsing. This parsing

will then be interpreted semantically to see if it is indeed the correct parsing. If not, we will assign the sentence its second most probable parsing and check again. We will continue to do this until we get the correct parsing for the sentence.

We can now see in what way our approach to sentence structure determination is better than the approaches of Hays and Simmons. Hays and Simmons produced all possible parsings for a sentence with no rank to them. Therefore each parsing was considered to have equal probability of being correct. If this equal probability approach were used in a question-answering system we would have to produce one answer for each parsing of the question. By using our approach we eliminate the need to consider all the parsings of a sentence thus saving time, and we will eliminate most of the irrelevant answers to a question.

Let us now look at some of the sentences which we have parsed with our program (Figure 12 a, b, c).

The	D Shape	
Eight-sided	D Shape	
Octagon	D Shape	
Shape	D Means	Means
Always	D Means	
Means	D	
Stop	D Means	Means

Figure 12a

A	D	Sign	
Stop	D	Sign	Sign
Sign	D	Is	
Is	D		
Red	D	Is	
With	D	Red	
White	D	Letters	
Letters	D	With	

Figure 12b

Signs	D	Posted	Like	
Like	D	Signs	Signs	Posted
This	D	Are	Are	
Are	D	Posted		
Posted	D			
On	D	Posted		
Main	D	Highways	On	
Highways	D	On		

Figure 12c

Figure 12

The sentences in Figure 12a and Figure 12b are unambiguously parsed and assigned the tree shown in Figure 13a and Figure 13b respectively

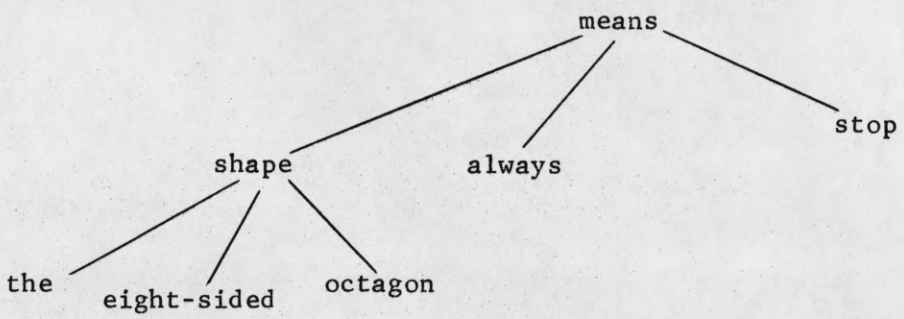

Figure 13a

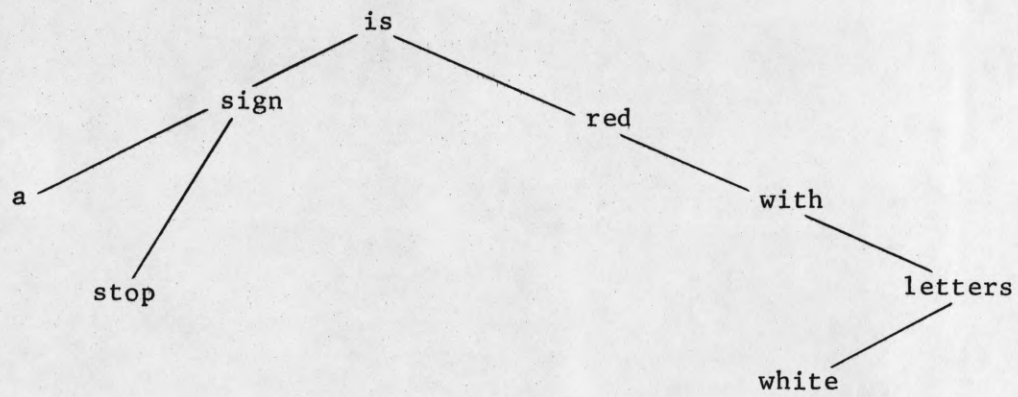

Figure 13b

The sentence of Figure 12c however is not unambiguously parsed since the program tells us that "signs" depends on either "posted" or "like." It will be noticed, however, that "this" is unambiguously

connected to the wrong governor. It is connected to the wrong governor because it finds the word "are" which is a possible governor and stops never getting to the word "like" which is the actual governor. We would like to point out though, that "this" is the only word in the sentence which is mis-connected due to our philosophy of parsing and that in most of the sample sentences which had faulty connections due to our method of parsing there was only one such faulty connection. Looking at the sentence in Figure 12c again, we also see that if we are allowed to make a second pass at the sentence, allowing "this" to look for its second most probable governor, the connection between "this" and "like" would be established and we would have the correct parsing of the sentence.

REFERENCES

1. Andreyev, N. D., "Linguistic Aspects of Translation," Proc. of 9th International Congress of Linguistics, Cambridge, Mass., 1962. Also Janua Linguarum, Series Maior, V. 12, 1962.

2. Bobrow, D. G. "Syntactic Theories in Computer Implementations," in Borko, ed., Automated Language Processing, New York, Wiley, 1967.

3. Chomsky, N. "Syntactic Structures" The Hague: Mouton and Co., 1957.

4. _____, "Three Models for the Description of Language." IRE Trans. on Information Theory, V.IT-2 (September, 1956).

5. Gaifman, H., "Dependency Systems and Phrase-Structure Systems," Information and Control, V. 8 (June, 1965), 304-337.

6. Hays, D. G. "Automatic Language-Data Processing" in Borko, ed., Computer Applications in the Behavioral Sciences, Englewood Cliffs, N. J.: Prentice-Hall, 1962.

7. _____, "Grouping and Dependency Theories," RM-2646. The Rand Corp., Santa Monica, California, September, 1960.

8. _____, "Dependency Theory: A Formalism and Some Observations," RM-4087-PR. The Rand Corp., Santa Monica, California, July, 1964.

9. _____, and Ziehe, T. W. "Studies in Machine Translation--10: Russina Sentence-Structure Determination," RM-2538. The Rand Corp., Santa Monica, California, April, 1960.

10. Kuno, S. "The Predictive Analyzer and a Path Elimination Technique" Comm. of ACM, V. 8 (July, 1965), 453-461.

11. _____, "The Augmented Predictive Analyzer for Context-Free Languages--Its Relative Efficiency" Comm. of ACM, V. 9 (November, 1966), 810-823.

12. _____, "Computer Analysis of Natural Languages" presented at the Symposia in Applied Mathematics, 1967.

13. McConlogue, K. and Simmons, R. F. "Analyzing English Syntax with a Pattern-Learning Parser," Comm. of ACM, V. 11 (November, 1965) 687-698.

14. Simmons, R. F. "Automated Language Processing" in Cuadra, ed., Annual Review of Information Science and Technology, 1965, New York, Interscience Publishers, 1966.

Distribution List as of November 1, 1968

1 Dr A.A. Dougal
Asst Director (Research)
Ofc of Defense Res & Eng
Department of Defense
Washington, D.C. 20301

1 Office of Deputy Director
(Research and Technology)
ODD R&E-OSD
The Pentagon, Room 3-E-144
Washington, D.C. 20301

1 Director Advanced Research Projects Agency
Department of Defense
Washington, D.C. 20301

1 Director for Information Sciences
Advanced Research Projects Agency
Department of Defense
Washington, D.C. 20301

1 Director for Materials Sciences
Advanced Research Projects Agency
Department of Defense
Washington, D.C. 20301

1 Headquarters
Defense Communications Agency (333)
The Pentagon
Washington, D.C. 20305

20 Defense Documentation Center
Attn: TISIA
Cameron Station, Bldg 5
Alexandria, Virginia 22314

1 Director
National Security Agency
Attn: Librarian C-332
Fort George G. Meade, Maryland 20755

1 Weapons Systems Evaluation Group
Attn: Col Daniel W. McElwee
Department of Defense
Washington, D.C. 20305

1 National Security Agency
Attn: R4-James Tippett
Office of Research
Fort George G. Meade, Maryland 20755

1 Central Intelligence Agency
Attn: OCR/DD Publications
Washington, D.C. 20505

1 Colonel Kee
AFRSTE
Hqs, USAF
Room ID-429, The Pentagon
Washington, D.C. 20330

1 Aerospace Medical Division
AMD (AMRXI)
Brooks Air Force Base, Texas 78235

1 AUL3T-9663
Maxwell AFB, Alabama 36112

1 AFFTC (FTBPP-2)
Technical Library
Edwards AFB, Calif. 93523

1 Hq SAMSO (SMITA/Lt Nelson)
AF Unit Post Office
Los Angeles, California 90045

1 Lt Col Charles M. Waespy
Hq USAF (AFRDSD)
Pentagon
Washington, D.C. 20330

1 SSD (SSTRT/Lt Starbuck)
AFUPO
Los Angeles, California 90045

1 Det #6, OAR (LOOAR)
Air Force Post Office
Los Angeles, California 90045

1 ARL (ARIY)
Wright-Patterson AFB, Ohio 45433

1 Dr H.V. Noble
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

1 Mr Peter Murray
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

1 AFAL (AVTE/R.D. Larson)
Wright-Patterson AFB, Ohio 45433

2 Commanding General
Attn: STEWS-WS-VT
White Sands Missile Range
New Mexico, 88002

1 RADC (EMLAL01)
Griffiss AFB, New York 13442
Attn: Documents Library

1 Mr H.E. Webb (EMIA)
Rome Air Development Center
Griffiss AFB, New York 13442

1 Academy Library (DFSLB)
U.S. Air Force Academy
Colorado Springs, Colorado 80912

1 Mr Morton M. Pavane, Chief
AFSC Scientific and Liaison Office
26 Federal Plaza
New York, N.Y. 10007

1 Lt Col Bernard S. Morgan
Frank J. Seiler Research Laboratory
U.S. Air Force Academy
Colorado Springs Colorado 80912

1 Technical Library, AFETR
(ETV, MU-135)
Patrick AFB, Florida 32925

1 AFETR (ETLLG-1)
STINFO Office (For Library)
Patrick AFB, Florida 32925

1 Dr L. M. Hollingsworth
AFCRL (CRN)
L.G. Hanscom Field
Bedford, Massachusetts 01731

1 AFCRL (CRMZLR)
AFCRL Research Library, Stop 29
L.G. Hanscom Field
Bedford, Mass 01731

1 Colonel Robert E. Fontana
Dept of Electrical Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

1 Colonel A.D. Blue
RTD (RTTL)
Bolling Air Force Base, D.C. 20332

1 Dr I.R. Mirman
AFSC (SCT)
Andrews AFB, Maryland 20331

1 AFSC (SCTR)
Andrews AFB, Maryland 20331

1 Lt Col J.L. Reeves
AFSC (SCBB)
Andrews AFB, Maryland 20331

2 ESD (ESTI)
L.G. Hanscom Field
Bedford, Mass 01731

1 AEDC (ARO, INC)
Attn: Library/Documents
Arnold AFS, Tenn 37389

2 European Office of Aerospace Research
Shell Building
47 Rue Cantersteen
Brussels, Belgium

5 Lt Col Robert B. Lalisch
Chief, Electronics Division
Directorate of Engineering Sciences
Air Force Office of Scientific Research
Arlington, Virginia 22209

1 APGC (PGBS-12)
Elgin AFB, Florida 32542

1 U.S. Army Research Office
Attn: Physical Sciences Division
3045 Columbia Pike
Arlington, Virginia 22204

1 Research Plans Office
U.S. Army Research Office
3045 Columbia Pike
Arlington, Virginia 22204

1 Commanding General
U.S. Army Materiel Command
Attn: AMCRD-TP
Washington, D.C. 20315

1 Commanding General
U.S. Army Strategic Communication Command
Fort Huachuca, Arizona 85613

1 Commanding Officer
Army Materials & Mech. Res. Center
Watertown Arsenal
Watertown, Mass. 02172

1 Commanding Officer
U.S. Army Ballistics Research Laboratory
Attn: AMXRD-BAT
Aberdeen Proving Ground
Aberdeen, Maryland 21005

1 Commandant
U.S. Army Air Defense School
Attn: Missile Sciences Division
C & S Dept
P.O. Box 9390
Fort Bliss, Texas 79916

1 Commanding General
U.S. Army Missile Command
Attn: Technical Library
Redstone Arsenal, Alabama 35809

1 U.S. Army Munitions Command
Attn: Technical Information Command
Picatinney Arsenal
Dover, New Jersey 07801

1 Commanding Officer
Harry Diamond Laboratories
Attn: Dr Berthold Altman (AMXDO-TI)
Connecticut Ave & Van Ness St. N.W.
Washington, D.C. 20438

1 Commanding Officer
U.S. Army Security Agency
Arlington Hall
Arlington, Virginia 22212

1 Commanding Officer
U.S. Army Limited War Laboratory
Attn: Technical Director
Aberdeen Proving Ground
Aberdeen, Maryland 21005

1 Commanding Officer
Human Engineering Laboratories
Aberdeen Proving Grounds
Aberdeen, Maryland 21005

1 Commandant
U.S. Army Command & General Staff
College
Attn: Secretary
Fort Leavenworth, Kansas 66270

1 Commanding Officer
U.S. Army Research Office (Durham)
Attn: CRD-AA-IP (Richard O. Ulsh)
Box CM, Duke Station
Durham, North Carolina 27706

1 Librarian
U.S. Army Military Academy
West Point, New York 10996

1 The Walter Reed Institute of Research
Walter Reed Medical Center
Washington, D.C. 20012

1 Commanding Officer
U.S. Army Electronics R & D Activity
White Sands Missile Range
New Mexico 88002

1 Dr H. Robl
Deputy Chief Scientist
U.S. Army Research Office (Durham)
Box CM, Duke Station
Durham, North Carolina 27706

1 U.S. Army Mobility Equipment
Research & Development Center
Attn: Technical Document Center
Bldg 315
Fort Belvoir, Virginia 22060

1 Mr Norman J. Field (AMSEL-RD-S)
Chief, Office of Science & Technology
U.S. Army Electronics Command
Fort Monmouth, New Jersey 07703

1 Mr Robert O. Parker
Executive Secretary,
JSTAC (AMSEL-XL-D)
Fort Monmouth, New Jersey 07703

1 Commanding General
U.S. Army Electronics Command
Fort Monmouth, New Jersey 07703
Attn: AMSEL-SC
RD-D
RD-G
RD-GF
RD-MAT
XL-D
XL-E (Dr K. Schwidta)
XL-C
1 Cy to each symbol listed. XL-S
HL-D
HL-CTR
HL-CT-P (Dr W. McAfee)
HL-CT-L
HL-CT-O
HL-CT-I
HL-CT-A
NL-D
NL-A
NL-P-2 (D. Haratz)
NL-R
NL-S
KL-D
KL-E
KL-S
KL-T
VL-F (R.J. Niemela)
WL-D

1 Director Night Vision Laboratory
U.S. Army Electronics Command
Attn: HL-NV-II (Dr A.D. Schnitzler
Fort Belvoir, Virginia 22060

1 Components Research Laboratory
(P.E. Landis) Bldg 92
Harry Diamond Laboratories
Connecticut Ave & Van Ness St N.W.
Washington, D.C. 20438

1 Mr Edward Vaughan
Research & Engineering Directorate
U.S. Army Weapons Command
Rock Island, Illinois 61201

1 Commanding General
U.S. Army Missile Command
AMSMI-REX (W. Todd)
Redstone Arsenal, Ala 35809

3 Chief of Naval Research
Department of the Navy
Washington, D.C. 20360
Attn: Code 427

2 Naval Electronics Systems Command
ELEX 03
Falls Church, Virginia 22046

1 Naval Ship Systems Command
SHIP 031
Washington, D.C. 20360

1 Naval Ships Systems Command
SHIP 035
Washington, D.C. 20360

2 Naval Ordnance Systems Command
ORD 32
Washington, D.C. 20360

2 Naval Air Systems Command
AIR 03
Washington, D.C. 20360

2 Commanding Officer
Office of Naval Research Branch Office
Box 39, Navy No.100 F.P.O.
New York, New York 09510

1 Commanding Officer
Office of Naval Research Branch Office
219 South Dearborn Stree
Chicago, Illinois 60604

1 Commanding Officer
Office of Naval Reserve Branch Office
207 West 24th Street
New York, New York 10011

1 Commanding Officer
Office of Naval Research Branch Office
1030 East Green Street
Pasadena, California 91101

1 Commanding Officer
Office of Naval Research Branch Office
495 Summer Street
Boston, Massachusetts 02210

8 Director, Naval Research Laboratory
Technical Information Officer
Washington, D.C. 20360
Attn: Code 2000

1 Commander
Naval Air Development & Material Center
Johnsville, Pennsylvania 18974

2 Librarian
U.S. Naval Electronics Laboratory
San Diego, California 95152

1 Commanding Officer & Director
U.S. Naval Underwater Sound Laboratory
Fort Trumbull
New London, Connecticut 06840

1 Librarian
U.S. Naval Post Graduate School
Monterey, California 93940

1 Commander
U.S. Naval Air Missile Test Center
Point Magu, California 93041

1 Director
U.S. Naval Observatory
Washington, D.C. 20390

2 Chief of Naval Operations
OP-07
Washington, D.C. 20350

1 Director, U.S. Naval Security Group
Attn: G43
3801 Nebraska Avenue
Washington, D.C. 20390

2 Commanding Officer
Naval Ordnance Laboratory
White Oak, Maryland 21502

1 Commanding Officer
Naval Ordnance Laboratory
Corona, California 91720

1 Commanding Officer
Naval Ordnance Test Station
China Lake, California 93555

1 Commanding Officer
Naval Training Device Center
Orlando, Florida 32811

1 Commanding Officer
Naval Avionics Facility
Indianapolis, Indiana 46241

1 U.S. Naval Weapons Laboratory
Dahlgren, Virginia 22448

1 Weapons Systems Test Division
Naval Air Test Center
Patuxtent River, Maryland 20670
Attn: Library

1 Head, Technical Division
U.S. Naval Counter Intelligence
Support Center
Fairmont Building
4420 North Fairfax Drive
Arlington, Virginia 22203

1 Mr Charles Yost
Special Asst to the Director of Res.
National Aeronautics & Space Admin.
Washington, D.C. 20546

1 Dr H. Harrison, Code RRE
Chief, Electrophysics Branch
National Aeronautics & Space Admin.
Attn: Library C3/TDL
Green Belt, Maryland 20771

1 NASA Lewis Research Center
Attn: Library
21000 Brookpark Road
Cleveland, Ohio 22135

1 National Science Foundation
Attn: Program Director
Engineering System Program ENG
1800 G. Street, N.W.
Washington, D.C. 20550

1 U.S. Atomic Energy Commission
Division of Technical Information Ext.
P.O. Box 62
Oak Ridge, Tenn. 37831

1 Los Alamos Scientific Laboratory
Attn: Reports Library
P.O. Box 1663
Los Alamos, New Mexico 87544

2 NASA Scientific & Technical Inform. Fac.
Attn: Acquisitions Branch (S/AK/DL)
P.O. Box 33
College Park, Maryland 20740

1 Director
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Mass 02139

1 Polytechnic Institute of Brooklyn
55 Johnson Street
Brooklyn, New York 11201
Attn: Mr Jerome Fox
Research Coordinator

1 Director
Columbia Radiation Laboratory
Columbia University
538 West 120th Street
New York, New York 10027

1 Director
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

1 Director
Stanford Electronics Laboratories
Stanford University
Stanford, California 94305

1 Director
Electronics Research Laboratory
University of California
Berkeley, California 94720

1 Director
Electronic Sciences Laboratory
University of Southern California
Los Angeles, California 90007

1 Electronics Research Center
University of Texas at Austin
Engr-Science Bldg 110
Austin, Texas 78712

1 Division of Engineering &
Applied Physics
210 Pierce Hall
Harvard University
Cambridge, Massachusetts 02138

1 Aerospace Corporation
P.O. Box 95085
Los Angeles, California 90045
Attn: Library Acquisition Group

1 Professor Nicholas George
California Inst. of Technology
Pasadena, California 91109

1 Aeronaurics Library
Graduate Aeronautical Laboratories
California Institute of Technology
1201 E. California Blvd
Pasadena, California 91109

1 Director, USAF Project Rand
Via: Air Force Liaison Office
The Rand Corporation
1700 Main Street
Santa Monica, California 90406

1 Hunt Library
Carnegie Institute of Technology
Schenley Park, Pittsburgh, Pa. 15213

1 Syracuse University
Dept of Electrical Engineering
Syracuse, New York 13210

1 Yale University
Engineering Dept
New Haven, Connecticut 06520

1 Airborne Instruments Laboratory
Deerpark, New York 11729

1 Bendix Pacific Division
11600 Sherman Way
North Hollywood, California 91605

1 General Electric Co
Research Laboratories
Schenectady, New York 12301

1 Lockheed Aircraft Corp
P.O. Box 504
Sunnyvale, California 94088

1 Raytheon Co
Bedford, Mass 01730
Attn: Librarian

1 Dr G. J. Murphy
The Technological Institute
Northwestern University
Evanston, Illinois 60201

1 Dr John C, Hancock, Director
Electronics Systems Research Laboratory
Purdue University
Lafayette, Indiana 47907

1 Director
Microwave Laboratory
Stanford University
Stanford, California 94305

1 Emil Schafer, Head
Electronics Properties Infor. Center
Hughes Aircraft Co
Culver, California 90230

1 The John Hopkins University
Applied Physics Laboratory
8621 Georgia Avenue
Silver Springs, Maryland 20910
Attn: Boris W. Kuvshinoff
Document Librarian

1 Dr Leo Young
Stanford Research Institute
Menlo Park, California 94025

1 Mr Henry Bachmann
Assistant Chief Engineer
Wheeler Laboratories
122 Cutterhill Road
Great Neck, New York 11021

1 School of Engineering Sciences
Arizona State University
Tempe, Arizona 85281

1 Engineering & Math Sciences Library
University of California at Los Angeles
405 Hilgred Avenue
Los Angeles, California 90024

1 California Institute of Technology
Pasadena, California 91109
Attn: Documents Library

1 University of California
Santa Barbara, California 93106
Attn: Library

1 Carnegie Institute of Technology
Electrical Engineering Dept
Pittsburgh, Pa 15213

1 University of Mighigan
Electrical Engineering Dept
Ann Arbor, Michigan 48104

1 New York University
College of Engineering
New York, N.Y. 10019

1 Dept of Electrical Engineering
Texas Technological College
Lubbock, Texas 79409

1 IBM Technical Info. Retrieval Center
International Business Machines Corp.
Armonk, New York 10504

1 Commander
Test Command (TCDT-E)
Defense Atomic Support Agency
Sandia Base
Albuquerque, New Mexico 87115

1 Commanding General
U.S. Army Weapons Command
Rock Island, Ill 61201
Attn: ANSWE-RDR (Gerald Reinsmith)

1 Col E.P. Gaines, ACDA/FO
1901 Pennsylvania Ave, N.W.
Washington, D.C. 20451

1 Mr Billy Locke
Plans Directorate
USAF Security Service
Kelly Air Force Base, Texas 78241

1 W.A. Eberspacher
Technical Consultant
Systems Integration
Code 5340A, Box 15
Naval Missile Center
Point Magu, California 93041

1 Director of Faculty Research
Department of the Air Force
USAF Academy
Colorado 80840

1 Lt Col Richard Bennett
AFRDD
The Pentagon
Washington, D.C. 20301

1 Weapons Systems Evaluation Group
Attn: Col John B. McKinney
410 Army-Navy Drive
Arlington, Virginia 22202

1 Mr M. Zane Thornton
National Library of Medicine
8600 Rockville Pike
Bethesda, Maryland 22014

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
University of Illinois Coordinated Science Laboratory Urbana, Illinois 61801	Unclassified
	2b. GROUP

3. REPORT TITLE

SYNTACTIC ANALYSIS FOR R2 SYSTEM

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

BISS, Kenneth

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December, 1968	34	14

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
DAAB-07-67-C-0199; also in part OE C-1-7-017213-4557.	R-399
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Distribution of this report is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Joint Services Electronics Program thru U.S. Army Electronics Command Ft. Monmouth, New Jersey 07703

13. ABSTRACT

The R2 system is being designed to automatically answer questions on the Illinois Driver's Manual--Rules of the Road. That is, the R2 system will accept any question based on Rules of the Road and, through some search procedure, pick the answer to the question. At some point in the search it will be necessary to syntactically analyze the input question or some data text, or both. It is for this reason that we initiated the investigation which has led to this report.

DD FORM 1 NOV 65 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Syntactic Analysis						
Automatic Sentence Structure Determination						
Automatic Sentence Parsing						
A Dependency Grammar Parsing Program						
Review of Dependency Grammars						